My problem # 1: bridge between logic and neural

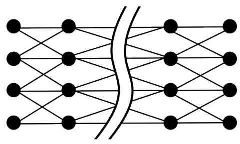
甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. 如何将**神经网络**和**逻辑规则**的结构结合，在混合的泛函空间上做 gradient descent？

1 神经网络的结构

一个 神经网络 F 基本上是：

$$\boldsymbol{y} =$$  $$(\boldsymbol{x})$$

$$F(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L\ \boldsymbol{x}))) \tag{1}$$

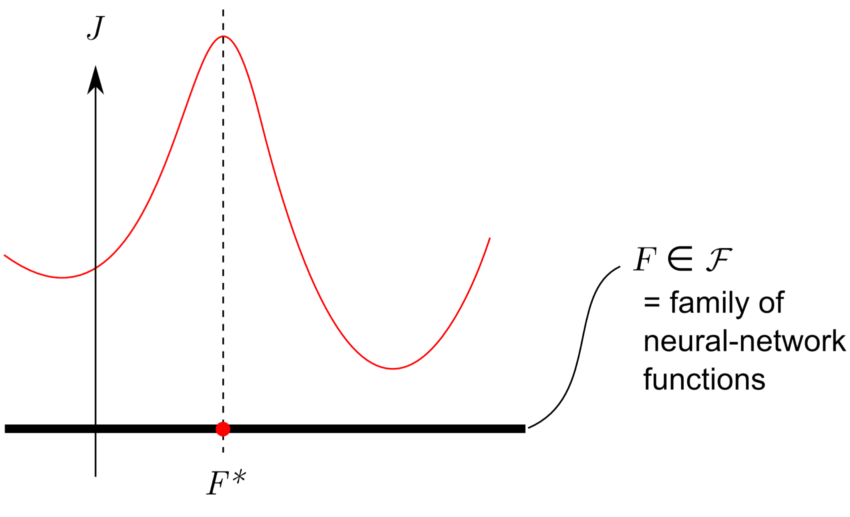
为简单起见，输入 $\boldsymbol{x}$ 和输出 $\boldsymbol{y}$ 都是 n-dim vector。
L 是层数，W_ℓ 是每层的**权重**矩阵 ($= n \times n$ square matrices)。
σ 是 sigmoid function （其作用是赋予非线性）：

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} \tag{2}$$

applied **component-wise**。必要时 σ 可以换成 $\mathbb{C}$ 上的 polynomials。

可以证明，如果层数和 neuron 个数足够多，这些 F functions 的 family $\mathcal{F}$ 在某个 function space 上是 dense 的。

Machine learning 的目的是在这 function space 上「计分」(ie, distribute scores $\in \mathbb{R}$ over functions $F \in \mathcal{F}$):



(3)

学习是透过 gradient descent 找出分数 J 最优的 F^*。方法是：

$$w \mathrel{+}= \eta \, \nabla_w \, J \tag{4}$$

其中 $w = W^{\ell}_{ij}$ 是第 ℓ 层的第 i, j 个权重，$\nabla_w J = \frac{\partial J}{\partial w}$ 就是 gradient，$\eta \ll 1$ 是控制学习速度的 learning rate。

~~可以将$\mathcal{F}$ 看成是由 和W's generate 出来的 Banach algebra（乘法是 composition）~~。Banach algebra 和 C^*-algebra 在此处不能应用，因为 不是线性算子。

2 逻辑结构

重点是想在 $\mathcal{F}$ 的结构上「加入」逻辑 rules 的结构。逻辑是指由一些命题推导出另一些命题：

$$A, B, C, ... \vdash D \tag{5}$$

这里的 $\vdash$ 的作用 corresponds to F。

逻辑结构有两部分：

- **Matching** structure: the state variable $\boldsymbol{x}$ is the Cartesian product of a number of smaller states $\boldsymbol{x}_i$. "Matching" means to check if $\boldsymbol{x}_i$ resides in certain polytope regions:

$$\begin{aligned} \text{if } A\boldsymbol{x}_i \geq \boldsymbol{b} \ \text{ then } \ &\mathbf{1} \\ \text{else } \ &\mathbf{0} \end{aligned} \tag{6}$$

  This is the same as applying the step function after a matrix:

$$\boldsymbol{y} = (W \, \boldsymbol{x}) \tag{7}$$

  and this can be easily incorporated in the 神经网络's F if we allow to replace . Finally, 这个 matching 的结果要 multiply with 下面的 linkage function。

- **Linkage** structure: Consider this logic formula and notice the linkages between variables on the left and right sides: （爸爸的爸爸是爷爷）

$$\forall X \; \forall Y \; \forall Z. \quad \text{grandfather}(X,Z) \longleftarrow \text{father}(X,Y) \wedge \text{father}(Y,Z) \tag{8}$$

  In the state space $\mathbb{X} \ni \boldsymbol{x}$, linkage means that F projects the i-th coordinate of $\boldsymbol{x}$ directly to the j-th coordinate of $\boldsymbol{y}$. In other words,

$$f : (x_1, ...x_i, ...x_n) \mapsto (y_1, ..., (y_j = x_i), ...y_n) \tag{9}$$

  I use the notation $F_{(i,j)...}$ to denote that F contains the linkage from coordinate i to coordinate j, and so on.

3 我的问题

怎样将以上的两种结构 “combine” 在一起？

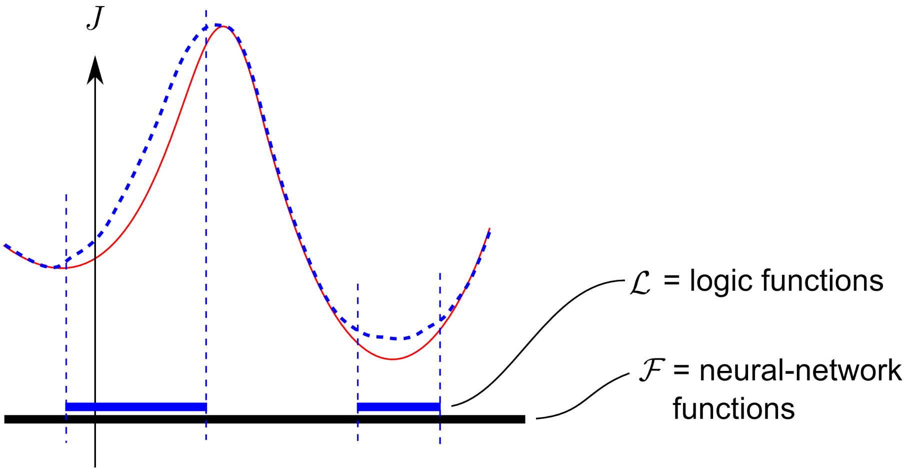
- $F \in \mathcal{F} =$ 所有神经网络函数的 family
- $L \in \mathcal{L} =$ 所有逻辑函数的 family （例如那些具有 linkage structure 的 functions）

目的是在这 “combined family” 上做 gradient descent。

$\mathcal{F}$ 的 closure 是 dense 的，所以 $\overline{\mathcal{F}} \supset \mathcal{L}$。

但 $\mathcal{F} \not\supseteq \mathcal{L}$。例如只要目测就可以看到 $\mathcal{F} \not\ni$ identity function，只能**近似**它。

形象化地看，$\mathcal{L}$ 是 $\mathcal{F}$ 的子集（注意 x-axis 代表 function space）：



(10)

我想做的是：在 function space 中搜寻时，将 $\mathcal{L}$ 中的元素加高 priority，换句话说 $L \in \mathcal{L}$ 的分数 $J(L)$ 较高（图中的蓝色虚线）。

注意：如果 F 有 linkage，那就变成一些 equational constraints over W. 每个 F 用参数 W^{ℓ}_{ij} 表示，$W \in \mathbb{R}^{\ell \times n \times n} = \mathbb{R}^m$ 。Linkage constraints 造成了 $\mathbb{R}^m$ 里的 **variety**。

有朋友建议用 Lagrangian multiplier，换句话说，maximize $J(x)$ subject to $g(x) = 0$. 但注意这个 $x \in \mathcal{F}$ 是函数空间，而且，对於每个 linkage，$g(x) = 0$ 的 constraint 是不同的。这个办法似乎不可行。

有个 algebra 的想法是：将 $\mathcal{F}$ 和 $\mathcal{L}$ 分解成一些 decompositions，然后将这些 factors 混合？但这个 idea 很含糊，我不知是不是这样做的。